

Compressão de Sinais Multimedia

Compressão minimiza o tamanho dos dados, tirando partido de redundâncias estatísticas, temporais, espaciais, espectrais, estruturais e perceptíveis.

- Critérios:
 - Taxa de compressão: $T_c = \frac{D_o}{D_c}$
 - Velocidade de processamento
- Compressão sem Perdas (Lossless)
 - Os dados originais são recuperados após a compressão
 - Taxa de compressão baixa
 - Usa-se em digitalização de documentos, obras de arte, medicina, c
 - Ficheiros: zip, rar, ...
- Compressão com Perdas (Lossy)
 - Há perda de informação
 - Perceptualmente esta perda é controlada
 - Taxa de compressão elevada
 - Usado em filmes, músicas, etc

Métodos de compressão sem perdas

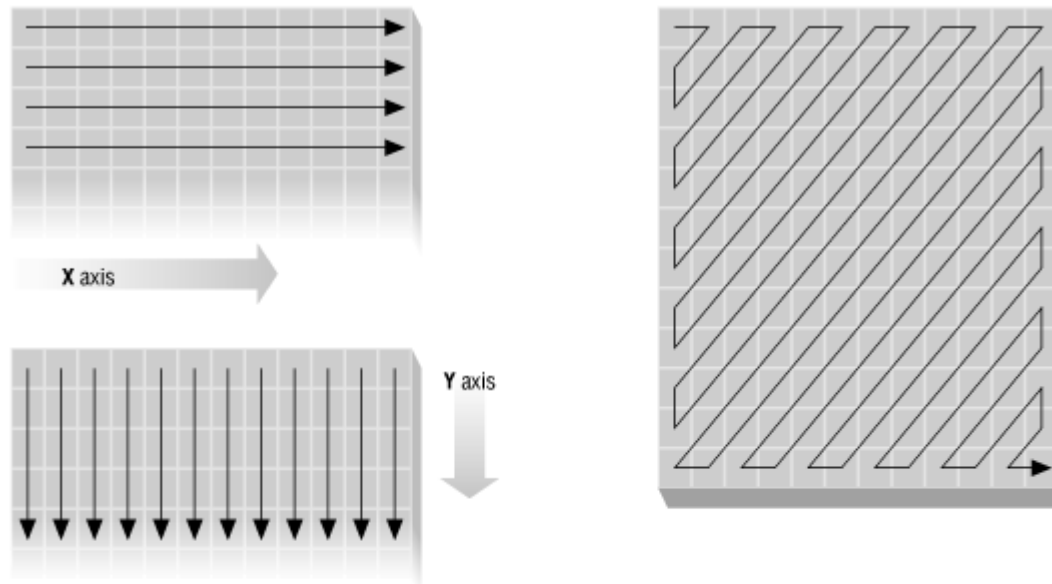
- Run Length Coding (RLC)
- Variable Length Coding (VLC)
 - Algoritmo Shannon-Fano
 - Código Huffman
 - Código de Huffman adaptativo
- Codificação Aritmética
- Codificação Baseada em dicionário
 - LZ77
 - LZ78
 - Lempel Ziv Storer Szymanski (LZSS)
 - Lempel Ziv Welch (LZW)
- Codificação Diferencial, Codificação Preditiva

Run Length Coding (RLC)

- Método de compressão suportado nos formatos BMP e TIFF;
- Baseia-se na redundância de informação contida nos dados;
- RLC substitui uma sequência de valores repetidos (run) por dois valores:
 - Run count – conta o número de repetições
 - Run value – o valor repetido
- Exemplo:
AAAAAAbbbCCCCCcd
6A3b6C1d
- Repare-se que 16 símbolos são representados por apenas 8 códigos!
- No caso da letra “d” há uma expansão.

Run Length Coding (RLC)

- Aplicação a sinais uni-dimensionais, ex: supressão de troços silêncio
- A aplicação deste método em imagens a preto e branco (ex: fax ou página de um livro) atinge níveis de compressão mais elevados que em imagens a cores.
- Variantes do método:



Run Length Coding (RLC)

- Bit-level: eficiente em imagens a preto e branco
 - o MSB de cada byte é o “run value” e
 - o “run count” varia entre 1 e 127
- Byte-level:
 - Primeiro byte é o “run count” (0-255)
 - Segundo byte é o “run value”
- Pixel-level:
 - Primeiro byte é o “run count” (0-255)
 - 3 bytes para o valor do pixel (ex: RGB)
- se o “run count” atinge o seu limite?

Codificação Entrópica de Sinais Multimedia

- Existe algum limite de compressão sem perdas óptimo que não possa ser melhorado?

Esta questão é respondida pela Teoria da Informação.

- É de bom senso que valores que ocorram muitas vezes tenham um código mais pequeno e os que ocorrem menos vezes código maior.
- Esta noção tem por base o estudo estatístico dos dados (histograma)

International Morse Code

1. A dash is equal to three dots.
2. The space between parts of the same letter is equal to one dot.
3. The space between two letters is equal to three dots.
4. The space between two words is equal to seven dots.

A	• —	U	• • —
B	— • • •	V	• • — —
C	— • — •	W	• — — —
D	— • •	X	— • • —
E	•	Y	— • — —
F	• • — •	Z	— — • •
G	— — •		
H	• • • •		
I	• •		
J	• — — —		
K	— • —		
L	• — • •		
M	— —		
N	— •		
O	— — —		
P	• — — •		
Q	— — • —		
R	• — •		
S	• • •		
T	—		
		1	• — — — —
		2	• • — — —
		3	• • • — —
		4	• • • • —
		5	• • • • •
		6	— • • • •
		7	— — • • •
		8	— — — • •
		9	— — — — •
		0	— — — — —

Conceitos básicos de Teoria de Informação

- Considerando uma fonte de informação, sem memória, S cujo alfabeto é $S = \{s_1, s_2, \dots, s_N\}$, onde os N símbolos são estatisticamente independentes,

- A informação própria é:

$$I(s_i) = \log_2 \left(\frac{1}{p(s_i)} \right) \quad [\text{bit}]$$

representa a quantidade de bits necessários para representar o símbolo s_i

- A entropia da fonte é:

$$\begin{aligned} H(S) &= \mathbf{E}[\log_2(1/p(s_i))] \\ &= - \sum_{i=1}^N p(s_i) \log_2 p(s_i) \quad [\text{bit/símbolo}] \end{aligned}$$

Mede a quantidade de informação codificada na mensagem.

- Nota: a entropia é tanto maior quando maior a incerteza.

Teorema da Codificação de fonte

- Shannon's noiseless source coding theorem:

É possível codificar uma fonte, sem perdas, com um número médio de bits por símbolo maior ou igual à entropia da fonte:

$$H(S) < L < H(S) + \delta$$

- Codificação entrópica, consiste em codificar uma fonte por forma a atingir um número médio de bits por símbolo igual à entropia dessa fonte.
- Eficiência do código:

$$\eta = \frac{H(S)}{L}$$

onde, L é o número médio de bits por símbolo

$$\begin{aligned} L &= \mathbf{E}[L(s_i)] \\ &= \sum_{i=1}^N p(s_i) L(s_i) \end{aligned}$$

Conceitos básicos de Teoria de Informação

- A entropia está compreendida no intervalo:

$$0 \leq H(S) \leq \log_2(N)$$

- Exemplo: Imagem a preto e branco com $p_1 = p_2 = 1/2$

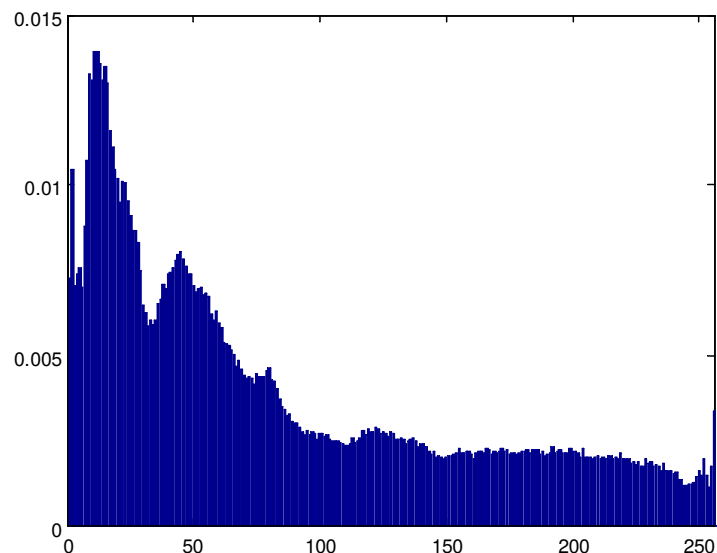
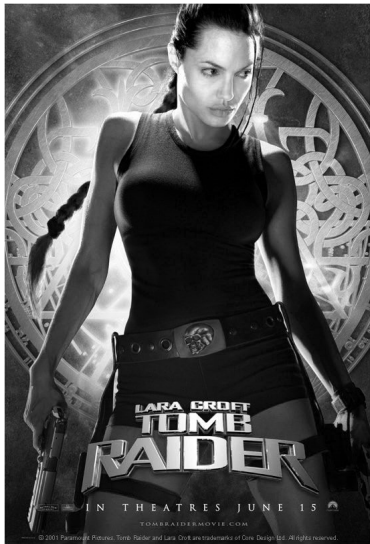
$$H(S) = ?$$

- Exemplo: fonte com 4 símbolos $p_1 = 5/8, p_2 = 1/8, p_3 = 1/8, p_4 = 1/8$,

$$H(S) = ?$$

- Exemplo: Imagem codificada com 256 níveis de cinzento

$$H(S) = ?$$



Algoritmo Shannon-Fano

- Método de codificação entrópica
- Código de comprimento variável (VLC)
- Método top-down

- Algoritmo
 - 1 – ordenar os símbolos por ordem decrescente de probabilidade (contabilizando o número de ocorrências de cada símbolo na mensagem)
 - 2 – Separar em dois grupos os símbolos com aproximadamente o mesmo número de ocorrências
 - 3 – Atribuir a cada grupo o bit 0 e 1
 - 4 – voltar ao ponto dois até que cada grupo tenha apenas um símbolo

Algoritmo Shannon-Fano

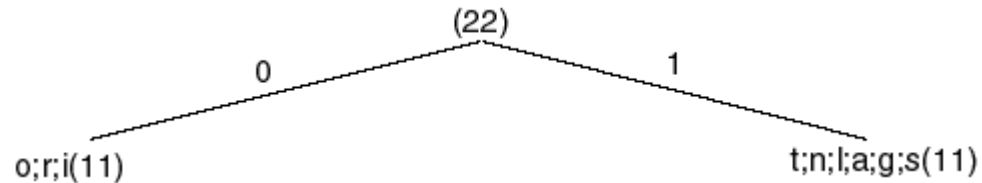
- Exemplo:
 - Mensagem: “otorrinolaringologista”
 - Símbolos / ocorrências

o	r	i	t	n	l	a	g	s
5	3	3	2	2	2	2	2	1

Algoritmo Shannon-Fano

- Exemplo:
 - Mensagem: “otorrinolaringologista”
 - Símbolos / ocorrências

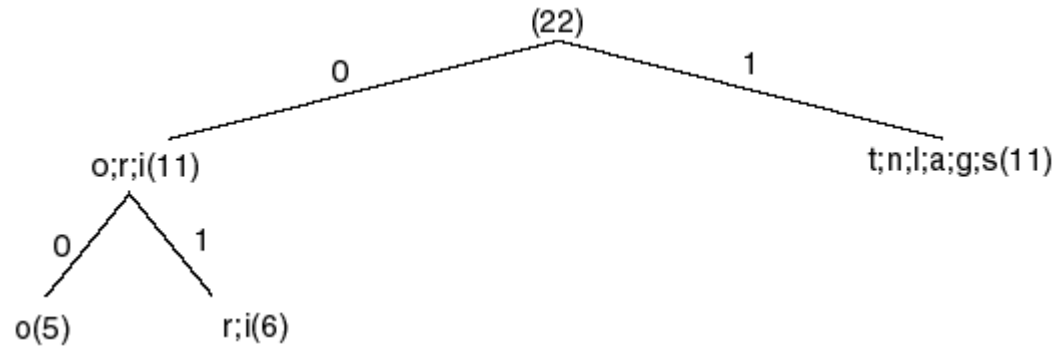
o	r	i	t	n	l	a	g	s
5	3	3	2	2	2	2	2	1



Algoritmo Shannon-Fano

- Exemplo:
 - Mensagem: “otorrinolaringologista”
 - Símbolos / ocorrências

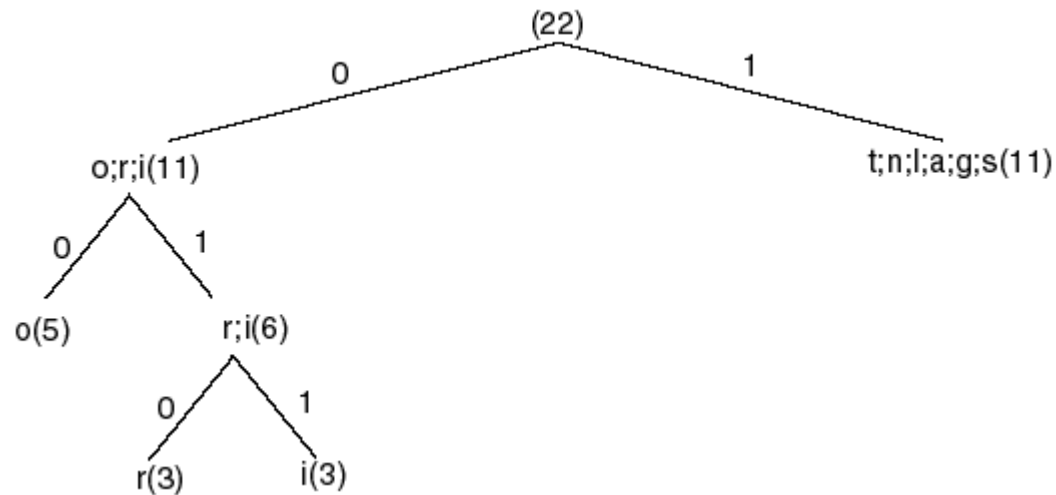
o	r	i	t	n	l	a	g	s
5	3	3	2	2	2	2	2	1



Algoritmo Shannon-Fano

- Exemplo:
 - Mensagem: “otorrinolaringologista”
 - Símbolos / ocorrências

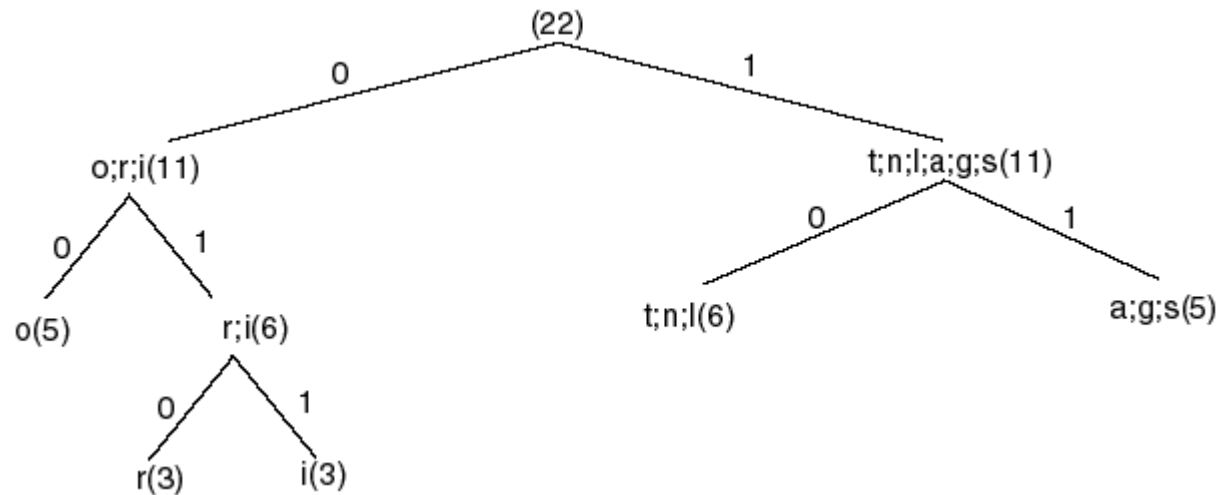
o	r	i	t	n	l	a	g	s
5	3	3	2	2	2	2	2	1



Algoritmo Shannon-Fano

- Exemplo:
 - Mensagem: “otorrinolaringologista”
 - Símbolos / ocorrências

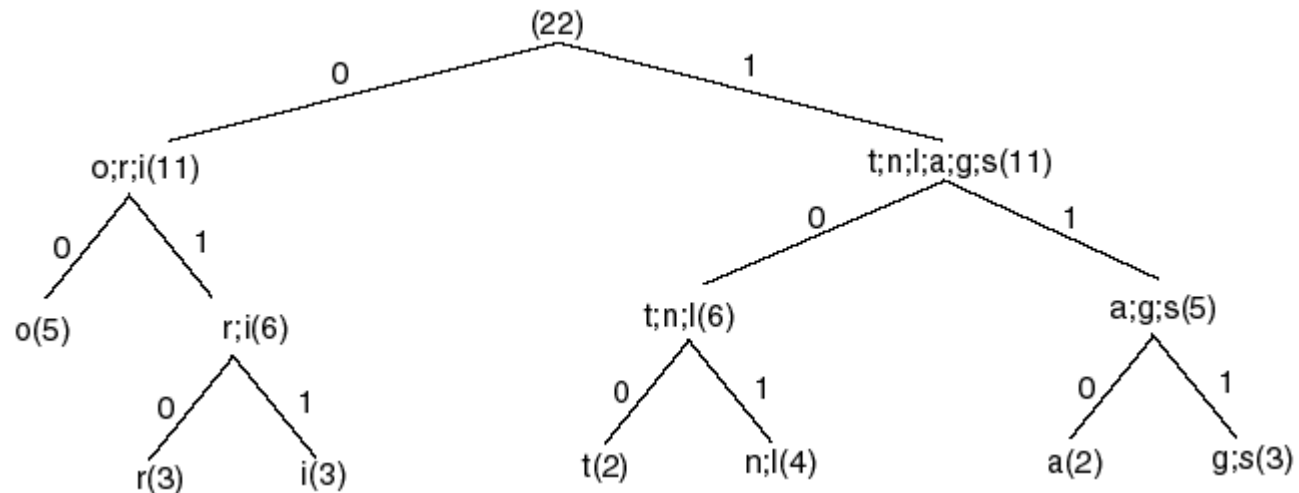
o	r	i	t	n	l	a	g	s
5	3	3	2	2	2	2	2	1



Algoritmo Shannon-Fano

- Exemplo:
 - Mensagem: “otorrinolaringologista”
 - Símbolos / ocorrências

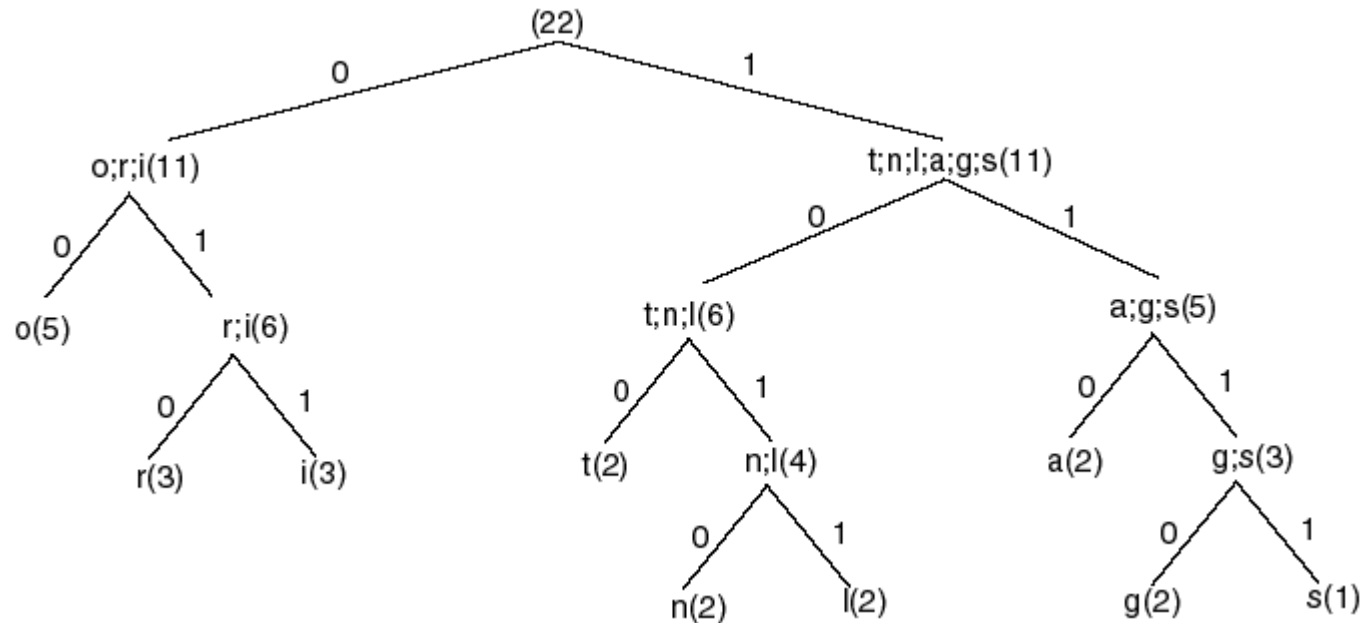
o	r	i	t	n	l	a	g	s
5	3	3	2	2	2	2	2	1



Algoritmo Shannon-Fano

- Exemplo:
 - Mensagem: “otorrinolaringologista”
 - Símbolos / ocorrências

o	r	i	t	n	l	a	g	s
5	3	3	2	2	2	2	2	1



Algoritmo Shannon-Fano

- Exemplo:
 - Mensagem: “otorrinolaringologista”
 - Código de cada símbolo

o	5	00
r	3	010
i	3	011
t	2	100
n	2	1010
l	2	1011
a	2	110
g	2	1110
s	1	1111

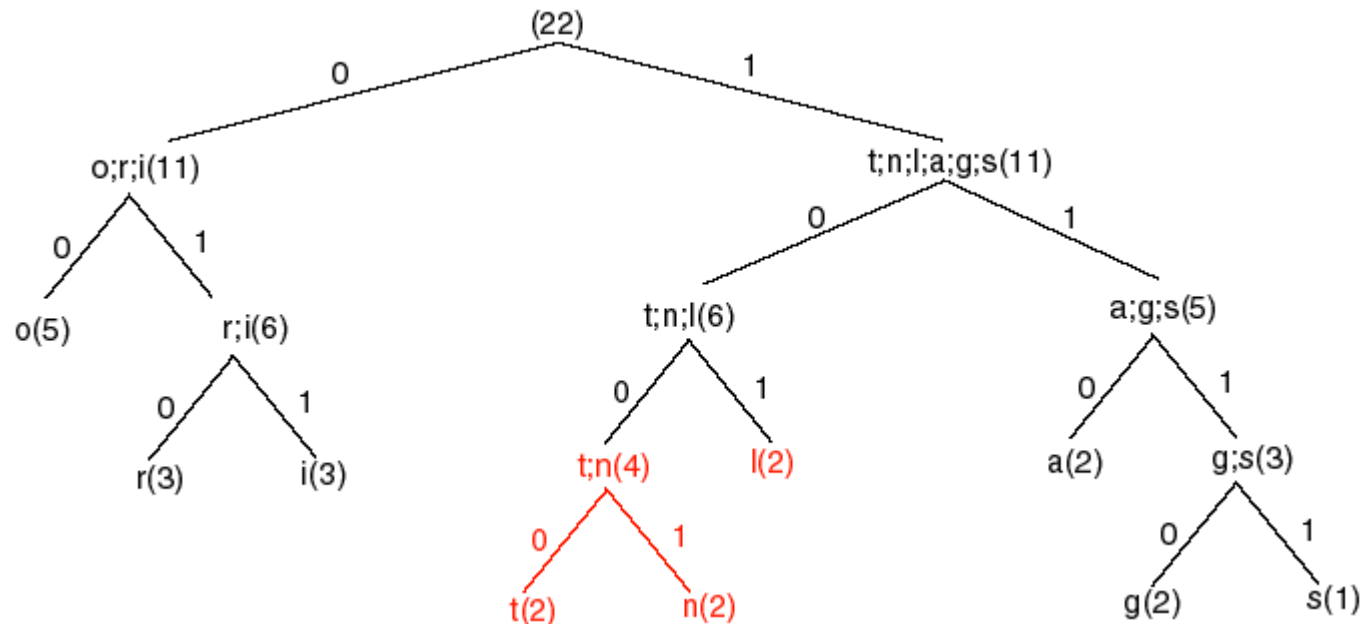
- Entropia = 3.045
- Número total de bits = 68
- Número médio de bits/ símbolo = 3.091
- Codificação com código de tamanho constante:
 - 4 bits (9 símbolos)
 - Total = 88 bits

- Mensagem: “00 100 00 010 010 011 1010 00 1011 110 ...”

Algoritmo Shannon-Fano

- Este algoritmo pode ter diferentes códigos para a mesma mensagem!
- Podendo alterar o número médio de bits por símbolo
- Exemplo:
 - Mensagem: “otorrinolaringologista”

o	r	i	t	n	l	a	g	s
5	3	3	2	2	2	2	2	1



Algoritmo Huffman

- Método de codificação entrópica
- Código de comprimento variável (VLC)
- Método bottom-up
- Usado na compressão JPEG e MPEG

- Algoritmo
 - 1 – Colocar os símbolos numa lista por ordem decrescente de probabilidade
 - 2 – escolher os dois símbolos com menor frequência e agrupá-los num novo símbolo com probabilidade igual à soma destes dois.
 - 3 – Inserir o novo símbolo na lista ordenada.
 - 4 – voltar ao ponto 2 até que cada a lista tenha apenas um símbolo.
 - 5 – Atribuir códigos partindo do ultimo símbolo criado.

- Nota: os códigos também se podem atribuir à medida que se cria o novo símbolo

Algoritmo Huffman

- Exemplo:
 - Mensagem: “otorrinolaringologista”

o (5)

r (3)

i (3)

t (2)

n (2)

l (2)

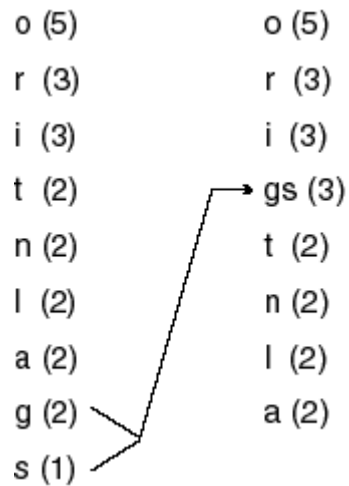
a (2)

g (2)

s (1)

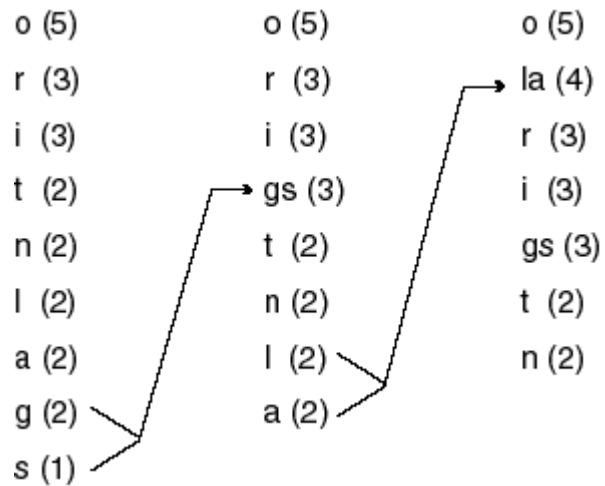
Algoritmo Huffman

- Exemplo:
 - Mensagem: “otorrinolaringologista”



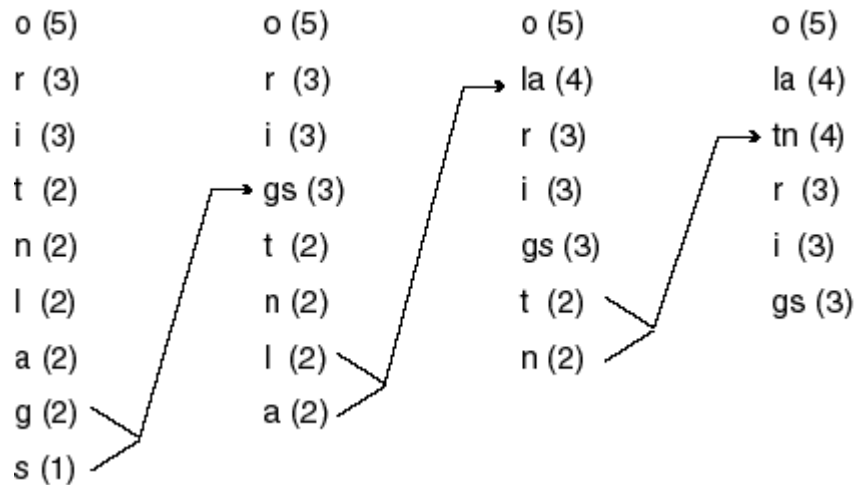
Algoritmo Huffman

- Exemplo:
 - Mensagem: “otorrinolaringologista”



Algoritmo Huffman

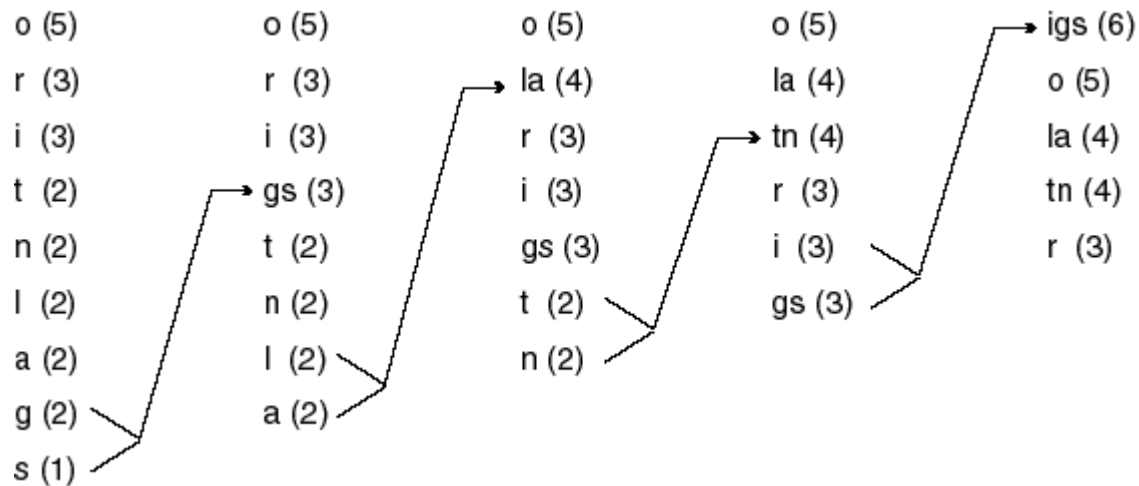
- Exemplo:
 - Mensagem: “otorrinolaringologista”



Algoritmo Huffman

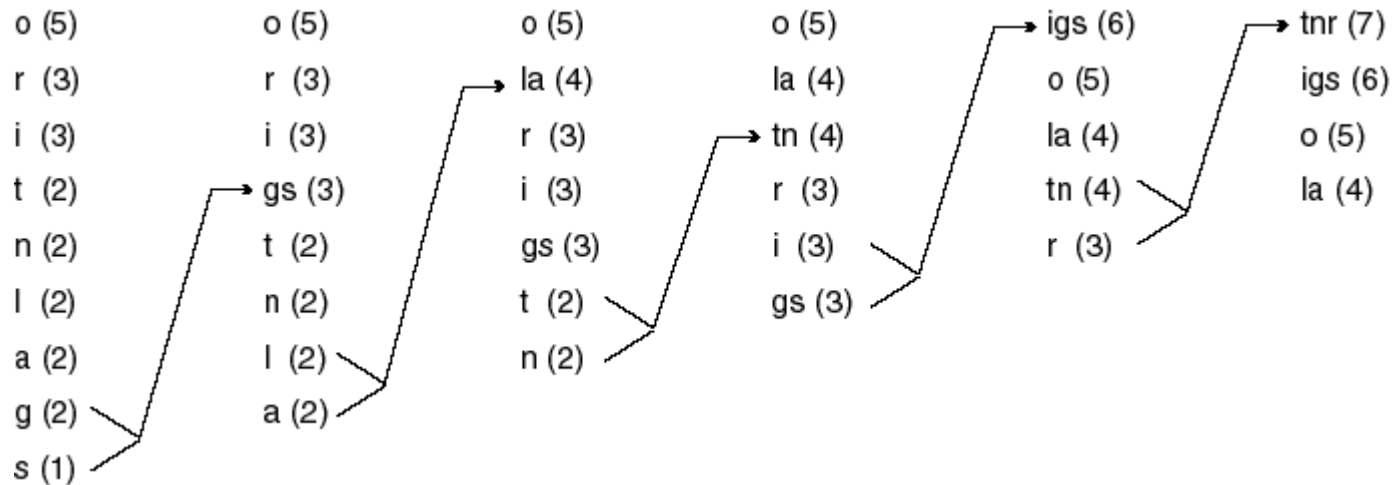
Exemplo:

■ Mensagem: “otorrinolaringologista”



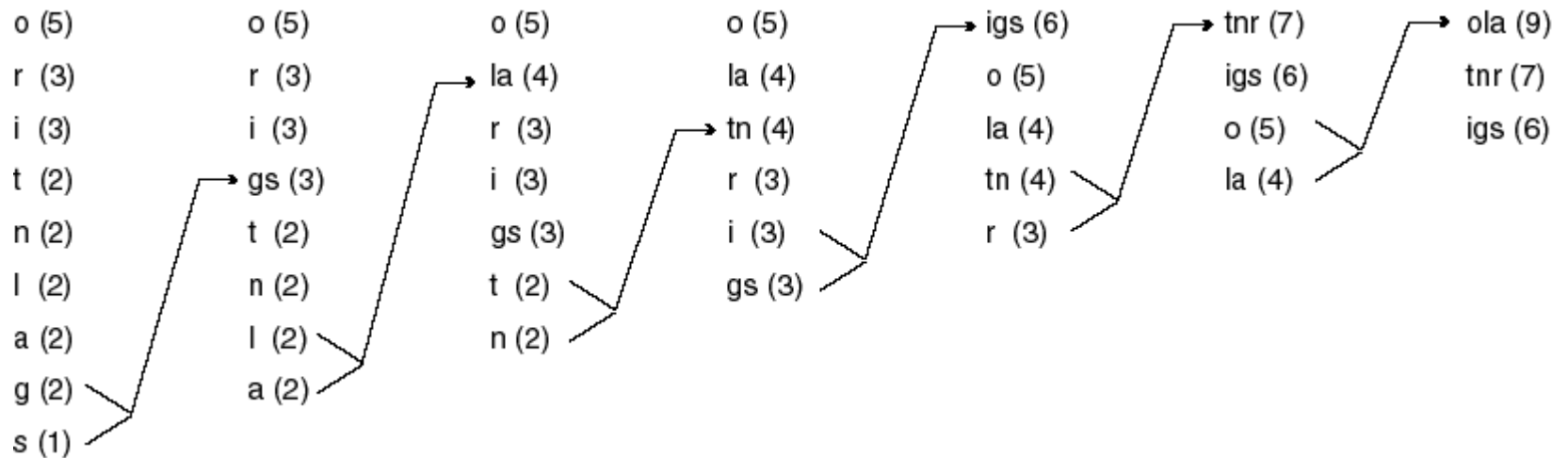
Algoritmo Huffman

- Exemplo:
 - Mensagem: “otorrinolaringologista”



Algoritmo Huffman

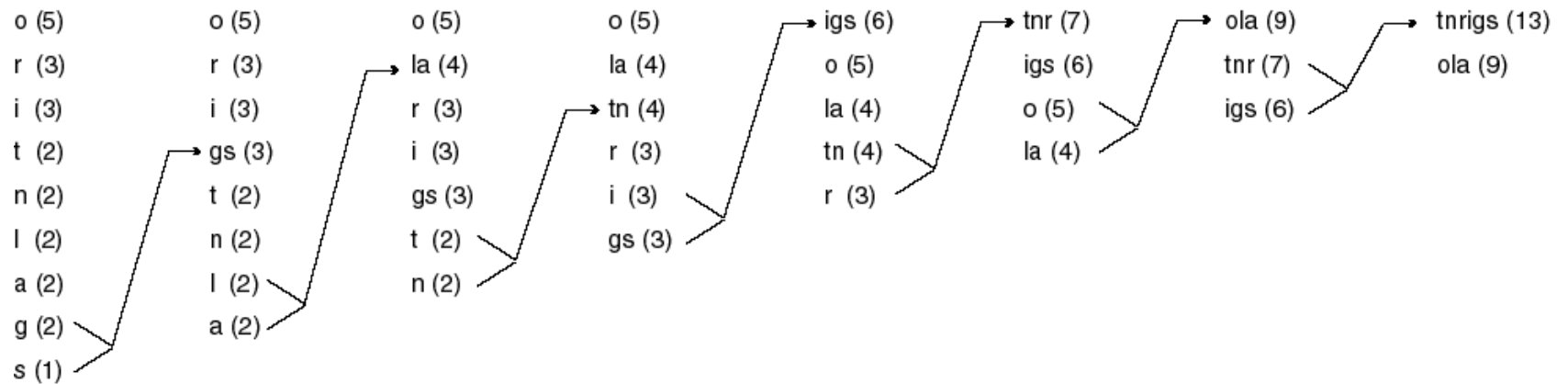
- Exemplo:
 - Mensagem: “otorrinolaringologista”



Algoritmo Huffman

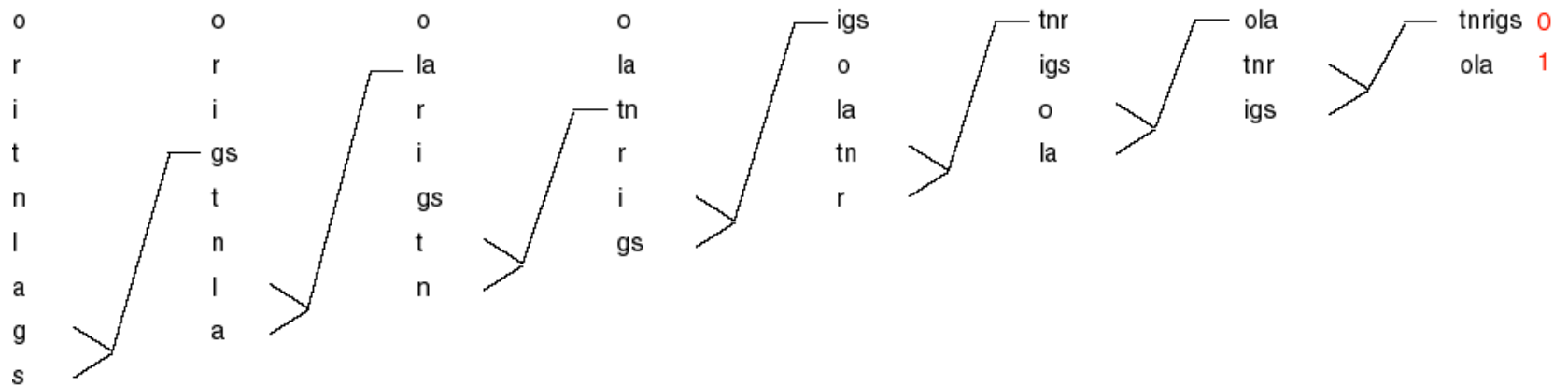
■ Exemplo:

■ Mensagem: “otorrinolaringologista”



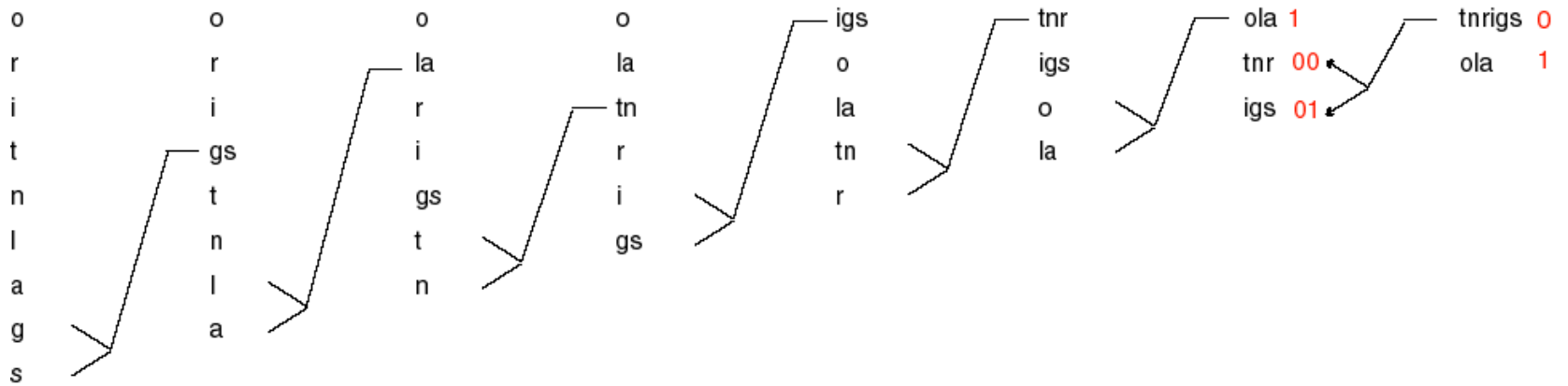
Algoritmo Huffman

- Exemplo:
 - Atribuição do código



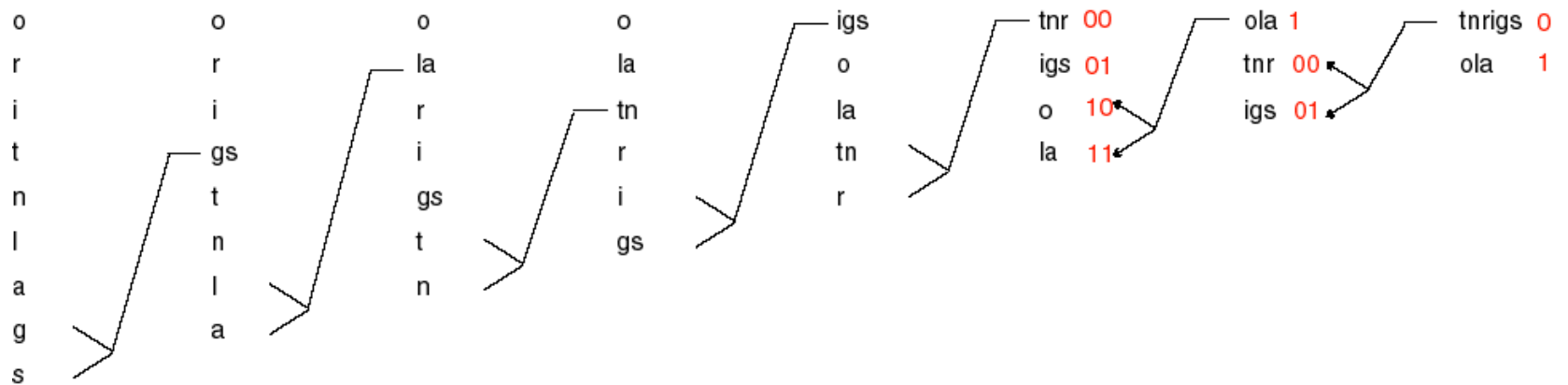
Algoritmo Huffman

- Exemplo:
 - Atribuição do código



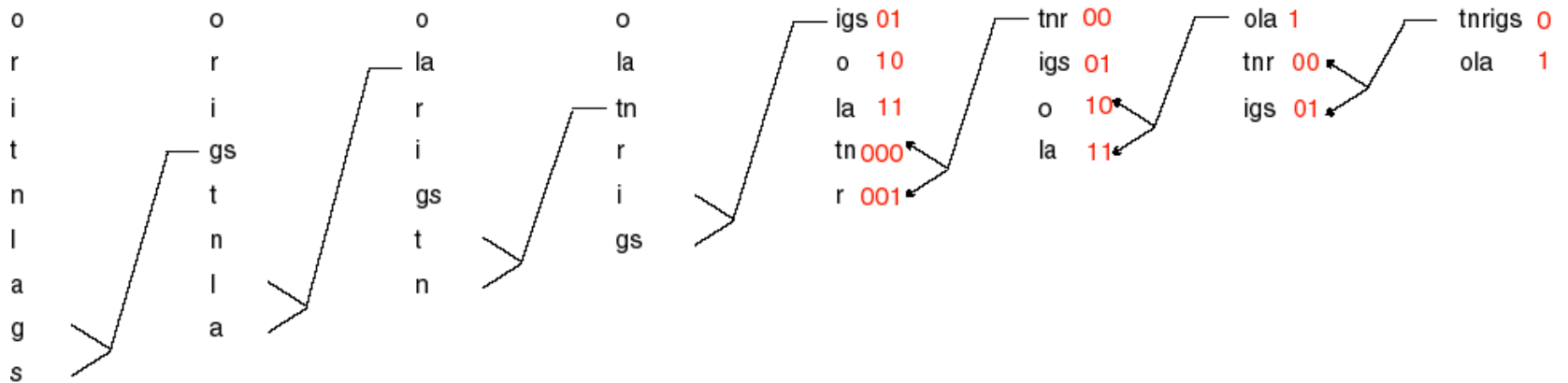
Algoritmo Huffman

- Exemplo:
 - Atribuição do código



Algoritmo Huffman

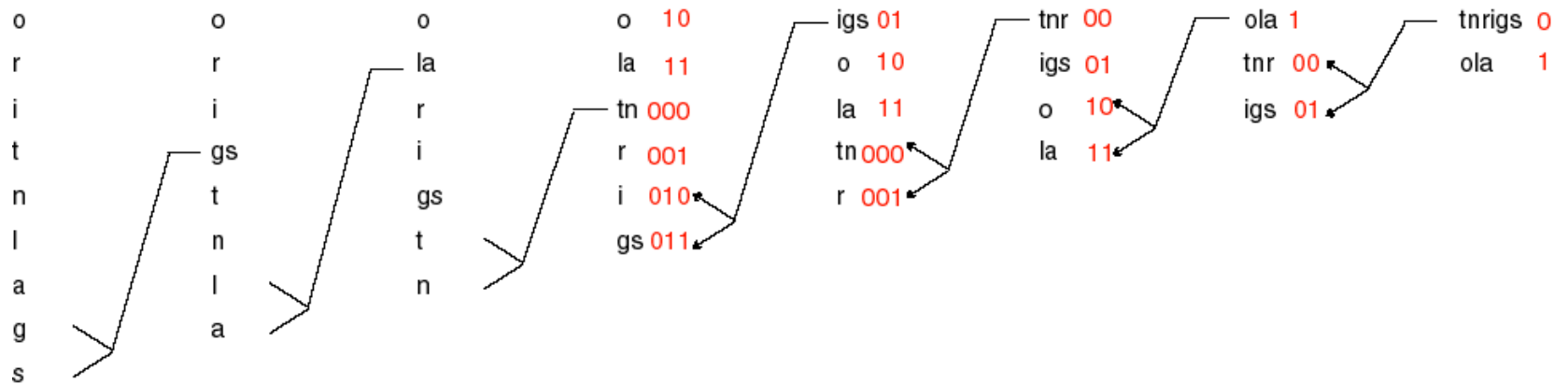
- Exemplo:
 - Atribuição do código



Algoritmo Huffman

Exemplo:

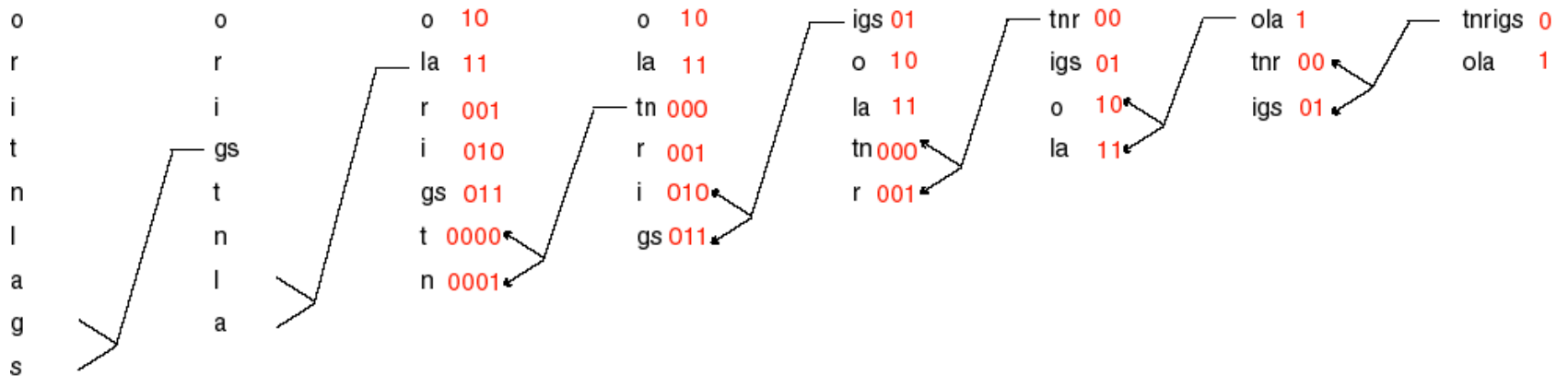
Atribuição do código



Algoritmo Huffman

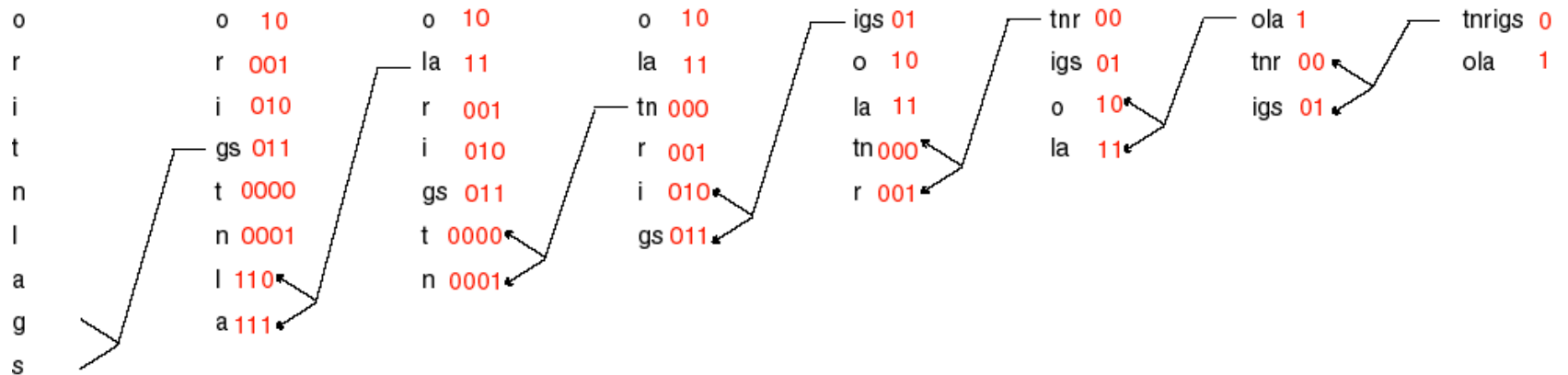
 Exemplo:

■ Atribuição do código



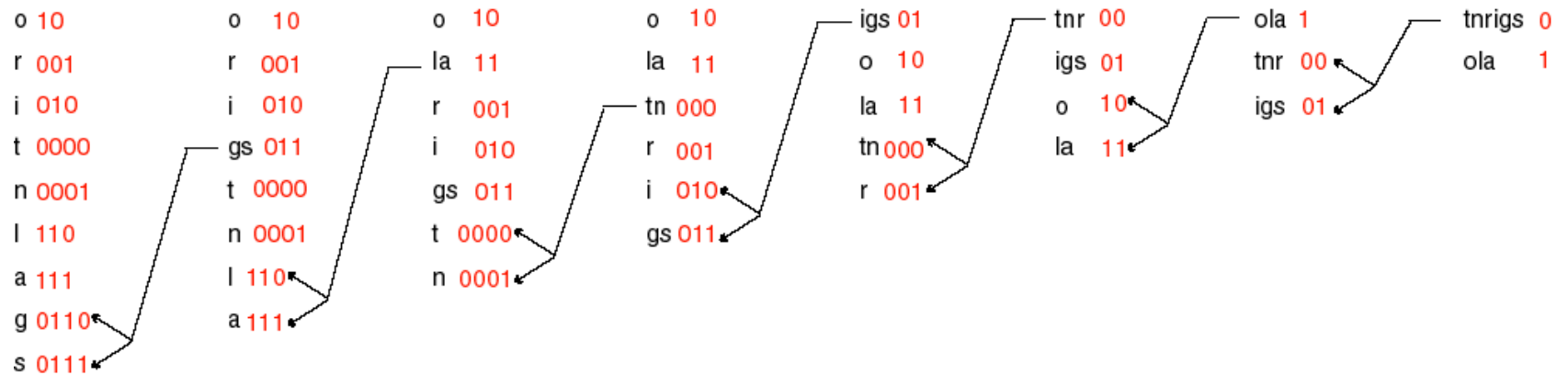
Algoritmo Huffman

- Exemplo:
 - Atribuição do código



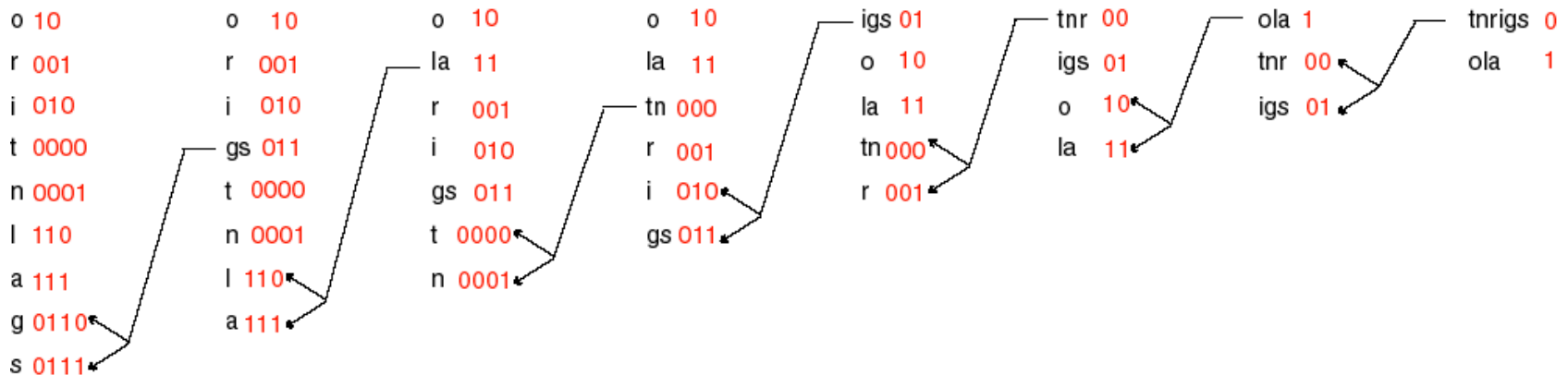
Algoritmo Huffman

- Exemplo:
 - Atribuição do código



Algoritmo Huffman

- Exemplo:
 - Atribuição do código



■ Mensagem: “10 0000 10 001 001 010 ...”

■ Total de bits = 68

- O código de Huffman é ótimo dado que garante:

$$H(S) \leq L < H(S) + 1$$

- Propriedade: nenhum código é prefixo de outro!

Algoritmo Huffman

- O código de Huffman:
 - Conhecimento à priori da densidade de probabilidade da fonte.
 - Esta informação tem de ser enviada no “header” do ficheiro
- No caso de sinais multimédia, muitas vezes não é possível obter este dado ou estima-lo com rigor.
- Código de Huffman adaptativo:
 - Inicialmente atribui-se um código a cada simbolo;
 - A estatística é actualizada sempre que se recebe mais um símbolo;
 - A árvore é actualizada, mantendo as suas propriedades (troca de nós)
 - Na mesma mensagem o mesmo simbolo tem diferentes códigos;

Código de Huffman

- Compressão de imagens:
 - Tipicamente divide-se a imagem em blocos de 8x8;
 - Cada bloco é um símbolo a codificar;
- No formato JPEG os blocos são primeiro transformados (DCT) e só depois se aplica o código de Huffman.